


[.NET Standard](#) [Advanced](#) [Beginner](#) [Intermediate](#) [.NET](#) [C#](#)

CliverRoutines

Sergiy Stoyan

★★★★★ 4.54/5 (4 votes)

Feb 3, 2021 [MIT](#) 3 min read [👁 5154](#)

Application settings manager. Logger. Auxiliary routines.

[Download source code from Github](#)

Introduction

CliverRoutines contains:

- application settings manager superseding .NET **ConfigurationManager**;
- logger with threading and session support;
- auxiliary routines;

CliverRoutines has been developed in .NET Standard 2.0 and is supposed to run on any platform adopting C# including .NET, Xamarin, Mono.

This article is a brief overview. For details, refer to [the documentation](#) and [the source code](#).

Background

Every time I was using .NET **ConfigurationManager**, it was a pain whether it was needed to store custom types, or initialize settings with long data, or override settings. In general, trying to make **ConfigurationManager** do something custom, throws into awkward coding full of restrictions. Not to mention that editing settings via the Visual Studio settings table is terribly unhandy.

That's why years ago, I developed the settings manager **Cliver.Config**. The idea behind it appeared so suitable that since then, I have never looked back. It has been used in many apps of any complexity on Windows and **Xamarin.Mac** and always proved to be simple in use, capable and robust.

Also, **CliverRoutines** exposes the logger **Cliver.Log** and some auxiliary routines.

Config

Cliver.Config is application settings manager intended as a replacement for .NET **ConfigurationManager**.

Features

- Cross-platform
- Enables setting types of any complexity and functionality
- Settings are easily modified directly in code
- Enables settings polymorphism
- Thread-safe
- Serializes data in JSON

Idea

Settings types are ordinary C# classes that you define in your code according to your needs thus achieving a great flexibility. **Cliver.Config** automatically detects fields/properties of those types in your code and facilitates their serialization/deserialization to/from disk.

While **Cliver.Config** was designed primarily as a settings manager, in conjunction with **System.Linq**, it can be used as a simple nosql database.

Usage

Reference **CliverRoutines** in your project.

Set your project's company name because it co-defines the storage directory.

Define settings types according to your needs. Generally, a settings type is an ordinary class that inherits from **Cliver.Settings** or one of its derivatives, e.g. **Cliver.UserSettings**.

Public non-static fields or properties in such a custom defined settings class will be automatically serialized/deserialized by command.

See the example of a settings type:

```
C#
//settings type
public class GeneralSettings : Cliver.UserSettings //serialized to the
//user-specific data directory
{
    public int Host; //it can be a property
    public int Port = 25;
    public List<Client> Clients = new List<Client>();
}

public class Client
{
    public string Name;
    public string Email;
}
```

Where you need it, declare a static field or property whose type is the defined settings type:

```
C#
public class Settings
```

```
{
    public static readonly GeneralSettings General; //readonly is optional.
                                                    //It can be a property.
}
```

Add this call at the beginning of your application, to make **Cliver.Config** detect all such fields/properties in your code and initialize them:

```
C#
//it can be called multiple times; every call re-initializes settings
Cliver.Config.Reload(); //mandatory call before using settings
```

Now settings are ready to be used:

```
C#
//modify
Settings.General.Host = "smtp.server.com";
Settings.General.Clients.Add(new Client {Name = "Tom", Email = "tom@company.com"});
...
//save on disk
Settings.General.Save();
//or, restore the previous values instead of saving the new ones
Settings.General.Reload();
//or, reset to the initial values
Settings.General.Reset();
...
//pass to a function
Client client = Settings.General.Clients.Find(a => a.Name == "Tom");
notify(Settings.General.Host, Settings.General.Port, client.Email);
```

Live examples can be found in **CliverRoutinesExample** project in **CliverRoutines** solution.
For the complete usage options, review **Cliver.Config** API.

Log

Cliver.Log is logger designed with usability in mind.

Features

- Cross-platform
- Thread-safe
- Session oriented - an application can write multiple log sessions successively or simultaneously. It is helpful when an application performs multiple independent tasks.
- Thread oriented - it can automatically write a log per thread
- Auto-cleanup of old logs
- Diagnostic output

Usage

Reference **CliverRoutines** in your project.

Set your project's company name because it co-defines the log directory.

At the beginning of the application, add optional initialization:

```
C#
using Cliver;
...

//it can be called many times; every call closes all the logs and re-initializes the engine.
Log.Initialize(Log.Mode.FOLDER_PER_SESSION); //this call is not mandatory
//if you are ok with default settings.
```

Trivial logging when everything is written to the same file:

```
C#
Log.Inform("test message");
```

Logging to a named log of an explicitly created session:

```
C#
//writing to the session "Game1"
Log.Session game1Session = Log.Session.Get("Game1");

//to the main log
game1Session.Warning("test message");

//to the log named "Test"
game1Session["Test"].Warning("test message");
```

In this example, each thread writes its own log:

```
C#
using Cliver;
...

//each download thread writes its own log
static void download(string uri)
{
    try
    {
        Log.Thread.Inform("test message");
    }
    catch (Exception e)
    {
        Log.Thread.Error2(e);
    }
}

static void Main(string[] args)
{
    //launch a number of parallel theads
    ThreadRoutines.Start(() => { download("http://file.com/1"); });
    ...
    ThreadRoutines.Start(() => { download("http://file.com/N"); });
}
```

Live examples can be found in **CliverRoutinesExample** project in **CliverRoutines** solution.
For complete usage options, review **Cliver.Log** API.

Enjoy!

History

- 3rd February, 2021: Initial version

